UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/789,201	02/27/2004	Christopher W. Brumme	MS1-1771US	9134

22801 7590 05/16/2007

LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

EXAMINER	
NGUYEN, PHILLIP H	

ART UNIT	PAPER NUMBER
2191	

NOTIFICATION DATE	DELIVERY MODE
05/16/2007	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

lhptoms@leehayes.com

Office Action Summary	Application No.	Applicant(s)	
	10/789,201	BRUMME ET AL.	
	Examiner	Art Unit	
	Phillip H. Nguyen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *27 February 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-31 and 39-54* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-31 and 39-54* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *27 February 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *20040227*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the original filing of February 27, 2004. Claims 1-31 and 39-54 are pending and have been considered below.

Note

2. Claim 1 recites the phrase "capable of" in the body of the claim. It indicates intended use and as such does not carry patentable weight. The limitations following the phrase "capable of" describe only intended use but not necessarily required functionality of the claim. In order for the limitations to be considered, Applicant is required to amend the claims so that the claim limitations are recited in a definite form. For example, claim 1 recites "capable of rewriting" should be changed to "rewrites" or any other definite form that does not indicate intended use.

Claim Rejections - 35 USC § 101

3. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

4. Claims 1-25 ~~and 51-54~~ P.N are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter:

- Regarding claims 1-25 recite a computer readable media, which is disclosed as signals. The specification provides intrinsic evidence that the computer readable media is intended to cover modulated data signals (e.g., carrier wave, RF, acoustic, infrared and other wireless media), such are currently not believed to

enable the computer readable media to act as a computer hardware component and realize its functionality absent being claimed in combination with the necessary hardware to received and convert the modulated signals to data structures. Therefore these claims are non statutory.

Claim Rejections - 35 USC § 112

5. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. Claim 1-6, 10-11, 22, 24-25, 29, 39-40 ~~and 50-54~~ are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

- Regarding to claim 1 recites, "one or more rewriters that include the at least one rewriter, each rewriter capable of rewriting the code unit" it is unclear to Examiner as to whether "at least one rewriter" and "each rewriter" are referred to "one or more rewriters" or to "at least one rewriter" of the rewriter list. For examining purposes, Examiner interprets this limitation as "wherein the at least one rewriter is capable of rewriting the code unit". Claims 2-6 directly or indirectly depend on claim 1, and therefore, suffer the same deficiency.
- Regarding claims 4-6, 22, 24-25, 29 and 39-40 recite "the group" in the preamble of the claim. There is insufficient antecedent basis for this limitation in the claim. For examining purposes, Examiner assumes "select from one of the recited lists".

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. Claims 1-11, 13, 14, 17, 18, 20, 21, 23, 25-29, 31, 40. 41 and 47-51 are rejected under 35 U.S.C. 102(b) as being anticipated by Lowney et al. (United States Patent No.: US 6,324,689 B1).

As per claim 1:

Lowney discloses:

- a code unit having executable instructions (see at least col. 3, line 64-65 "**executable files (shown labeled EXE)... are software application**"; also see FIG. 3);
- a rewriter list identifying at least one rewriter (see at least col. 4, line 4-5 "**pre-optimizer routine 25 and optimizer routine 30**"); and
- one or more rewriters that include the at least one rewriter, each rewriter capable of rewriting the code unit (see at least col. 4, line 5-6 "**pre-optimizer routine 25 optimizer routine 30 operate together to re-write the executables 21 and 22**").

As per claim 2:

Lowney discloses:

- a rewrite manager, the rewrite manager having executable instructions configured to access the at least one rewriter from the one or more rewriters and to execute the at least one rewriter against the code unit, generating a rewritten code unit (see at least col. 5, line 26-28 "**the pre-optimization tool works in conjunction with an optimization tool to re-write executable code**" – **pre-optimization tools can be considered as rewrite manager that works with optimization tool to re-write executable code**).

As per claim 3:

Lowney discloses:

- a cache, the rewrite manager further configured to store the rewritten code unit in the cache (see at least col. 3, line 60-61 "**during operation, as instructions and data of the software are executed...copied into the cache 16**").

As per claim 4:

Lowney discloses:

- a list of one or more custom attributes within the code unit, a list within a security policy, a list within an installation tool, a list within a configuration file, and a list within an XML (Extensible Markup Language) file (see at least **FIG. 1**; also see at

least col. 4, line 3-5 "**included in memory 18 are pre-optimizer routine 25 and optimizer routine 30**").

As per claim 5:

Lowney discloses:

- a stand-alone module; an operating system module; an execution environment module; a JIT compiler module; a source code compiler module; and an installation tool configured to install the code unit (see at least **FIG. 1**).

As per claim 6:

Lowney discloses:

- a developer computer configured to create the code unit; an intermediate computer configured to deploy the code unit; and a deployment computer configured to execute the code unit (see at least col. 3, line 49 "**a computer system**").

As per claim 7:

Lowney discloses:

- initiating a transformation of the code unit (**It is inherent in order to perform the re-arranging executable code**); and
- identifying one or more rewriters to implement the transformation (see at least col. 4, line 5-6 "**pre-optimizer routine 25 optimizer routine 30 operate together to re-write the executables 21 and 22**").

As per claim 8:

Lowney discloses:

- wherein the one or more rewriters is a plurality of rewriters, each rewriter configured to implement a unique transformation of the code unit (see at least col. 4, line 5-6 "**pre-optimizer routine 25 optimizer routine 30 operate together to re-write the executables 21 and 22" – meaning there is a plurality of rewriter routines**), the code unit including further executable instructions configured for sequencing the plurality of rewriters to implement the unique transformations in a particular order (**pre-optimizer routine is executed first for analyzing the type of the longwords and optimizer re-arranges the executable code**).

As per claim 9:

Lowney discloses:

- the code unit including further executable instructions configured for guiding the one or more rewriters to implement the transformation against identified elements within the code unit (see at least col. 5, line 7-14 "**each of the instructions and data that are included in the image 21 are formed from multi-bit words... a 32 bit words is hereinafter referred to as longwords" – pre-optimizer identifies the type of longwords for rewrite the executable code**).

As per claim 10:

Lowney discloses:

- wherein the initiating comprises: identifying an environment in which the transformation should be implemented (see at least col. 3, line 49-55 "**a computer system 10...includes a central processing unit (CPU)...memory 18...a cache 16... a processor 14...**"); and
- initiating the transformation only if the code unit experiences the identified environment (**the re-write process is performed for the computer system 10 is identified**).

As per claim 11:

Lowney discloses:

- identifying a source code compilation environment (see at least col. 3, line 49-55 "**a computer system 10...includes a central processing unit (CPU)...memory 18...a cache 16... a processor 14...**");
- identifying a pre-execution-compilation environment (see at least col. 3, line 49-55 "**a computer system 10...includes a central processing unit (CPU)...memory 18...a cache 16... a processor 14...**");
- identifying a compilation-on-installation environment (see at least col. 3, line 49-55 "**a computer system 10...includes a central processing unit (CPU)...memory 18...a cache 16... a processor 14...**");

- identifying an execution environment (see at least col. 3, line 49-55 "**a computer system 10...includes a central processing unit (CPU)...memory 18...a cache 16... a processor 14...**"); and
- identifying an installation environment (see at least col. 3, line 49-55 "**a computer system 10...includes a central processing unit (CPU)...memory 18...a cache 16... a processor 14...**").

As per claim 13:

Lowney discloses:

- wherein the identifying one or more rewriters comprises identifying a file separate from the code unit that includes a list of the one or more rewriters (see at least col. 4, line 3-4 "**included in memory 18 are pre-optimizer routine 25 and optimizer routine 30" – meaning they are not part of the executable code**).

As per claim 14:

Lowney discloses:

- receiving a code unit having executable instructions (see at least col. 3, line 58-59 "**software that executes on the computer system 10 is shown stored in the memory 18**");
- determining at least one rewriter with which the code unit may be rewritten (see at least col. 4, line 5 "**optimizer routine 30**");

- calling the at least one rewriter (see at least col. 5, line 41 "**the optimization or instrumenter routine is executed**"); and
- executing the at least one rewriter against the code unit to generate a rewritten code unit (see at least col. 5, line 44-45 "**optimization tool re-writes the executable...**").

As per claim 17:

Lowney discloses:

- computer-executable instructions configured for storing the rewritten code unit in a cache (see at least col. 3, line 60-61 "**during operation, as instructions and data of the software are executed...copied into the cache 16**").

As per claim 18:

Lowney discloses:

- receiving an instruction to execute the code unit (see at least col. 4, line 5-6 "**pre-optimizer routine 25 optimizer routine 30 operate together to re-write the executables 21 and 22**");
- recognizing that the code unit has been rewritten (see at least col. 3, line 60-61 "**during operation, as instructions and data of the software are executed...copied into the cache 16**" – **by recognizing the executed code has been rewritten, copying the executed code into the cache 16**);

- loading the rewritten code unit from the cache (see at least col. 3, line 60-61 "**during operation, as instructions and data of the software are executed...copied into the cache 16**"); and
- executing the rewritten code unit (see at least FIG. 1, **processor 14 is executing the code in cache 16**).

As per claim 20:

Lowney discloses:

- wherein the at least one rewriter is a plurality of rewriters (see at least col. 4, line 5-6 "**pre-optimizer routine 25 optimizer routine 30 operate together to re-write the executables 21 and 22**" – **This indicates a plurality of rewriter routines**), the one or more computer-readable media comprising further computer-executable instructions configured for sequencing the plurality of rewriters to rewrite the code unit in a particular rewrite order (**pre-optimizer routine is executed first for analyzing the type of the longwords and optimizer re-arranges the executable code**).

As per claim 21:

Lowney discloses:

- accessing a rewriter list (see at least col. 5, line 30 "**pre-optimization tool is executed", line 40 "optimization or instrumenter routine is executed**" – **accessing them by executing them**); and

- setting the rewrite order according to the rewriter list (**pre-optimization tool is executed first then optimization tool afterward**).

As per claim 23:

Lowney discloses:

- wherein the determining the at least one rewriter comprises reading a rewriter list (**It is inherent in order to use them because pre-optimizer and optimizer are stored in a memory**).

As per claim 25:

Lowney discloses:

- a developer computer configured to develop the code unit; an intermediate computer configured to install the code unit; and a deployment computer configured to execute the code unit (see at least col. 3, line 49 "**a computer system**").

As per claim 26:

Lowney discloses:

- a code unit (see at least col. 3, line 64-65 "**executable files (shown labeled EXE)... are software application**"; also see FIG. 3);
- a composable set of rewriters, each rewriter configured to rewrite the code unit in a unique manner (see at least col. 4, line 5 "**optimizer routine 30**");

- a rewrite manager configured to identify one or more rewriters from the composable set of rewriters and to execute the identified one or more rewriters against the code unit (see at least col. 5, line 26-27 "**the pre-optimization tool works in conjunction with an optimization tools to re-write executable code**" – **meaning optimization tool is selected to work with pre-optimization for re-write executable code**) ; and
- a rewritten code unit generated by executing the identified one or more rewriters against the code unit (see at least col. 5, line 44-45 "**optimization tool re-writes the executable ...**").

As per claim 27:

Lowney discloses:

- a rewrite cache, the rewrite manager further configured to store the rewritten code unit in the rewrite cache (see at least col. 3, line 60-61 "**during operation, as instructions and data of the software are executed...copied into the cache 16**").

As per claim 28:

Lowney discloses:

- a rewriter list from which the rewrite manager identifies the one or more rewriters to execute against the code unit (see at least col. 5, line 26-27 "**the pre-optimization tool works in conjunction with an optimization tools to re-write**

executable code" – meaning optimization tool is identified/selected to work with pre-optimization for re-write executable code).

As per claim 29:

Lowney discloses:

- a list of rewriters in the code unit; a list of rewriters in a stand-alone file; a list of rewriters in a security policy; and a list of rewriters in an installation tool (**optimization tool is stand-alone tool**).

As per claim 39:

Lowney discloses:

- a stand-alone rewrite module; a rewrite module configured as part of an operating system; a rewrite module configured as part of an installation tool; and a rewrite module configured as part of a security policy (**pre-optimization is stand-alone tool**).

As per claim 40:

Lowney discloses:

- a development computer configured to develop the code unit; an intermediate computer configured to install the code unit; and a deployment computer configured to execute the code unit (see at least col. 3, line 49 "**a computer system**").

As per claim 41:

Lowney discloses:

- receiving an executable code unit (see at least col. 3, line 58-59 "**software that executes on the computer system 10 is shown stored in the memory 18**");
- determining that the code unit needs to be rewritten (see at least col. 5, line 31-33 "**analyzes the executable to identify longwords as either instructions or data**");
- determining one or more rewriters to rewrite the code unit (see at least col. 5, line 26-27 "**the pre-optimization tool works in conjunction with an optimization tools to re-write executable code**" – **meaning optimization tool is determined/selected to work with pre-optimization for re-write executable code**); and
- running the one or more rewriters against the code unit to generate a rewritten code unit (see at least col. 5, line 44-45 "**optimization tool re-writes the executable ...**").

As per claim 44:

Lowney discloses:

- storing the rewritten code unit in a cache (see at least col. 3, line 60-61 "**during operation, as instructions and data of the software are executed...copied into the cache 16**").

As per claim 45:

Lowney discloses:

- receiving an instruction to execute the code unit (see at least col. 4, line 5-6 "**pre-optimizer routine 25 optimizer routine 30 operate together to re-write the executables 21 and 22**");
- recognizing that the rewritten code unit is stored in the cache (see at least col. 3, line 60-61 "**during operation, as instructions and data of the software are executed...copied into the cache 16**" – **by recognizing the executed code has been rewritten, copying the executed code into the cache 16**);
- loading the rewritten code unit from the cache (see at least col. 3, line 60-61 "**during operation, as instructions and data of the software are executed...copied into the cache 16**"); and
- executing the rewritten code unit (see at least FIG. 1, **processor 14 is executing the code in cache 16**).

As per claim 47:

Lowney discloses:

- sequencing the one or more rewriters to rewrite the code unit in a particular rewrite order (**pre-optimizer routine is executed first for analyzing the type of the longwords and then optimizer re-arranges the executable code**).

As per claim 48:

Lowney discloses:

- accessing a rewriter list (see at least col. 5, line 30 "**pre-optimization tool is executed", line 40 "optimization or instrumenter routine is executed" – accessing them by executing them**); and
- setting the rewrite order according to the rewriter list (**pre-optimization tool is executed first then optimization tool afterward**).

As per claim 49:

Lowney discloses:

- wherein the determining the at least one rewriter comprises reading a rewriter list (**It is inherent in order to use them because pre-optimizer and optimizer are stored in a memory**).

As per claim 50:

Lowney discloses:

- a developer computer configured to develop the code unit, an intermediate computer configured to install the code unit, and a deployment computer configured to execute the code unit (see at least col. 3, line 49 "**a computer system**").

As per claim 51:

Lowney discloses:

- receiving a code unit (see at least col. 3, line 58-59 "**software that executes on the computer system 10 is shown stored in the memory 18**");
- determining that the code unit is to be rewritten by a rewriter rewritten (see at least col. 5, line 31-33 "**analyzes the executable to identify longwords as either instructions or data**");
- determining if the code unit and the rewriter are trusted (**instructions and data of the software and optimization tool are always trusted software**);
- running the rewriter against the code unit to generate a rewritten code unit if the code unit and the rewriter are trusted (see at least col. 5, line 44-45 "**optimization tool re-writes the executable ...**"); and
- storing the rewritten code unit in a cache (the executable code was originally copied into the cache a (see at least col. 3, line 60-61 "**during operation, as instructions and data of the software are executed...copied into the cache 16**").

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. Claims 12, 22 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lowney et al. (United States Patent No.: US 6,324,689 B1).

As per claim 12:

Lowney does not explicitly discloses:

- listing the one or more rewriters within the code unit.

However, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to modify Lowney's approach to include the pre-optimization tool and optimization tool into the executable code. One would have been motivated to include pre-optimizer routine and optimizer routine into the executable code because it is one of the choices for performing the re-writing the executable code.

As per claim 22:

Lowney discloses:

- wherein the accessing comprises accessing the rewriter list in a location selected from the group comprising:
    - a separate file associated with the code unit (**pre-optimization tool and optimization tool are separated from the executable code**).

Lowney does not explicitly disclose:

- wherein the accessing comprises accessing the rewriter list in a location selected from the group comprising:

- the code unit, a separate file associated with the code unit, and a system policy.

However, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to modify Lowney's approach to include the pre-optimization tool and optimization tool into the executable code or system policy. One would have been motivated to include pre-optimizer routine and optimizer routine into the executable code because it is one of the choices for performing re-writing code.

As per claim 24:

Lowney discloses:

- a source code compiler (**pre-optimization routine and optimization routine are part of the compiler**);
- an installation tool (**pre-optimization routine and optimization routine are part of the installation tool in order to install the executable code after finished rewriting the code**);
- a managed execution environment (**pre-optimization and optimization routines are part of the managed execution environment**);
- a stand-alone rewrite management tool (**pre-optimization and optimization tool are stand-alone tool**).

Lowney does not explicitly

- a JIT (just in time) compiler.

However, it would have been obvious one having an ordinary skill in the art at the time the invention was made to recognize that Lowney's approach has a compiler for compiling code. One would have use JIT (just in time) complier in Lowney's approach because JIT is one of many choices available.

11. Claims 15, 16, 19, 30, 31, 42, 43, 46, 52 and 53 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lowney et al. (United States Patent No.: US 6,324,689 B1), in view of Cotugno et al. (United States Patent No.: US 6,353,887 B1).

As per claims 15 and 42:

Lowney does not explicitly disclose:

- computer-executable instructions configured for verifying trustworthiness of the code unit and the at least one rewriter prior to the executing.

However, Cotugno discloses an analogous way of distributing applications:

- computer-executable instructions configured for verifying trustworthiness of the code unit and the at least one rewriter prior to the executing (col. 20, line 53-56 "**One should not trust files that are supposedly wrapped with digital signatures under conditions when the system does not either attempt to verify the signature or fails to do so**").

Therefore, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to modify Lowney's approach to include a digital signature for verifying the files. One would have been motivated to include a digital signature for

verifying trustworthy of the files and verifying that the files have not been tampered or altered during it transmission (see Cotugno at least col. 20-21).

As per claims 16 and 43:

Cotugno discloses:

- wherein the verifying comprises authenticating a digital signature (see at least col. 18, line 20-21 "**authenticate the signature and verifying that the file has not been tampered**...").

As per claims 19 and 46:

Lowney does not explicitly disclose:

- generating a digital signature for the rewritten code unit; and
- associating the digital signature with the rewritten code unit.

However, Cotugno discloses an analogous way of distributing files.

- generating a digital signature for the rewritten code unit (see at least col. 18, line 23 "**Digital signature "creation" is based on the data within a file and a private key**"); and
- associating the digital signature with the rewritten code unit (see at least col. 18, line 23-26 "**Digital signature "creation" is based on the data within a file and a private key digital signature "authentication" is based on the same data of the file and a corresponding public key**").

Therefore, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to modify Lowney's approach to include a digital signature for verifying the files. One would have been motivated to create a digital signature for verifying trustworthy of the files and verifying that the files have not been tampered or altered during it transmission (see Cotugno at least col. 20-21).

As per claim 30:

Lowney does not explicitly disclose:

- a first digital signature associated with the code unit; and
- a set of second digital signatures, each second digital signature associated with a specific rewriter from the composable set of rewriters;
    - wherein the rewrite manager is further configured to determine if the code unit is trusted based on the first digital signature, determine if each rewriter from the identified one or more rewriters is trusted based on a corresponding second digital signature from the set of second digital signatures, and execute the identified one or more rewriters against the code unit only if both the code unit and each rewriter from the identified one or more rewriters are trusted.

However, Cotugno discloses an analogous way of creating signature associates with a file for distribution.

- a first digital signature associated with the code unit (see at least col. 18, line 14-15 "**a digital signature allows the recipient of a signed wrapped file to be certain of its original**" – **a file contains digital signature**); and
- a set of second digital signatures, each second digital signature associated with a specific rewriter from the composable set of rewriters (see at least col. 20, line 53-55 "**One should not trust files that are supposedly wrapped with digital signatures**..." – **Meaning each file contains a digital signature for authenticating of the file**);
    - wherein the rewrite manager is further configured to determine if the code unit is trusted based on the first digital signature, determine if each rewriter from the identified one or more rewriters is trusted based on a corresponding second digital signature from the set of second digital signatures, and execute the identified one or more rewriters against the code unit only if both the code unit and each rewriter from the identified one or more rewriters are trusted (see at least col. 20, line 53-58 "**one would not trust files that are supposedly wrapped with digital signatures under conditions when the system does not either attempt to verifying the signature or fails to do so**").

Therefore, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to modify Lowney's approach to include a signature in executable files, pre-optimizer routine and optimizer routine. One would have been motivated to include a digital signature in each one of them in order to allow the

recipient of a signed wrapped file to be certain of its origin and has not been tampered or altered during the transmission (see Cotugno at least col. 18, line 15-22).

As per claim 31:

Cotugno discloses:

- a third digital signature associated by the rewrite manager with the rewritten code unit and configured to verify that the rewritten code unit is trusted (see at least col. 18, line 20-22 "**authenticate the signature and verifying that the file has not been tampered or altered during its transmission**").

As per claim 52:

Lowney does not explicitly disclose:

- generating a digital signature for the rewritten code unit; and
- attaching the digital signature to the rewritten code unit.

However, Cotugno discloses:

- generating a digital signature for the rewritten code unit (see at least col. 18, line 23 "**Digital signature "creation" is based on the data within a file and a private key**"); and
- attaching the digital signature to the rewritten code unit (see at least col. 18, line 23-26 "**Digital signature "creation" is based on the data within a file and a private key digital signature "authentication" is based on the same data of the file and a corresponding public key**").

Therefore, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to modify Lowney's approach to include a digital signature for verifying the files. One would have been motivated to create a digital signature for verifying trustworthy of the files and verifying that the files have not been tampered or altered during it transmission (see Cotugno at least col. 20-21).

As per claim 53:

Lowney discloses:

- receiving a call to execute the code unit (see at least col. 3, line 58-59 "**software that executes on the computer system 10 is shown stored in the memory 18**");
- recognizing that the code unit has been rewritten (see at least col. 3, line 60-61 "**during operation, as instructions and data of the software are executed...copied into the cache 16**" – **by recognizing the executed code has been rewritten, copying the executed code into the cache 16**); and
- loading the rewritten code unit from the cache (see at least col. 3, line 60-61 "**during operation, as instructions and data of the software are executed...copied into the cache 16**");

However, Lowney does not explicitly disclose:

- verifying the digital signature attached to the rewritten code unit (see at least col. 18, line 20-22 "**authenticate the signature and verifying that the file has not been tampered or altered during its transmission**"); and

- executing the rewritten code unit if the verifying indicates that the rewritten code unit is secure (see at least col. 18, line 53-54 "**executed separately to make the file more secure**").

Therefore, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to modify Lowney's approach to include a digital signature for verifying the files. One would have been motivated to create a digital signature for verifying trustworthy of the files and verifying that the files have not been tampered or altered during it transmission (see Cotugno at least col. 20-21).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Phillip H. Nguyen whose telephone number is (571) 270-1070. The examiner can normally be reached on Monday - Thursday 10:00 AM - 3:00 PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Y. Zhen can be reached on (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

PN
5/7/2007

WEI ZHEN
SUPERVISORY PATENT EXAMINER